La session a été enregistrée en vidéo.

Les notes sont complémentaires à cet enregistrement vidéo, pas forcément autonomes

Feedback apprécié : http://eric.daspet.name/
Surtout si vous êtes en désaccord, ne comprenez pas, trouvez l'enchaînement mal fait, etc.

J'ai découvert initialement les API par du SOAP, du RPC et du « fait maison ».
Venant du web, j'ai été pas mal dégouté au départ.

Je suis un peu de l'autre côté de la barrière, à des documentations complètes, validations, transactions et ACID, je préfère la simplicité et des interfaces réduites. Un peu la même différence qu'entre un Oracle et un noSQL.

Chacun ses usages, mais on se tire plus facilement dans le pied avec un fusil chargé (SOAP).

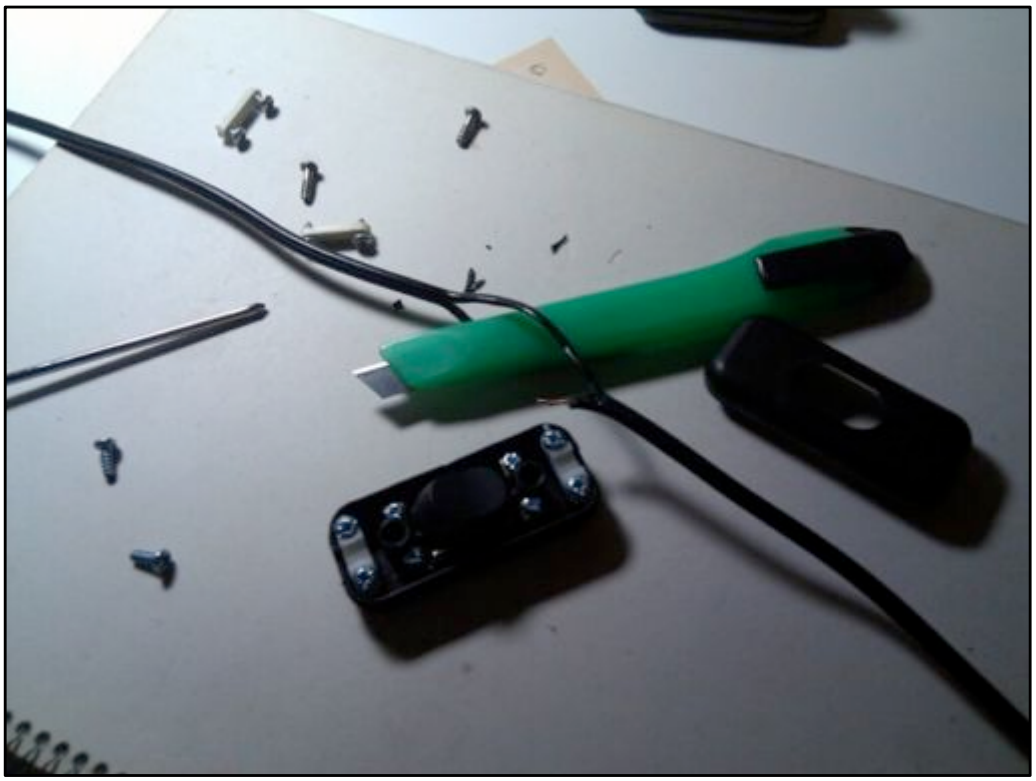
 # Web, REST

Les cool kids parlent de HTTP, REST, Hypermedia, HATEOAS ;
À chercher sur Wikipedia, il y a plein de présentations en ligne.

Je suis en plein dans cette philosophie, et ce sera ma direction

Par contre seuls quelques rares personnes ont vraiment raison d'y aller à fond
Ils feront progresser l'état de l'art, ce qui est très bien. Peut être qu'ils ouvriront la voie pour que HATEOAS soit l'avenir de tout le monde.
Mais entre temps vous avez probablement intérêt à prendre des raccourcis et ne pas aller jusqu'au bout du chemin, simplement parce que votre temps sera mieux investi sur votre métier réel.

**
Photo http://www.flickr.com/photos/joits/439556240 par Joits sous CC-NC



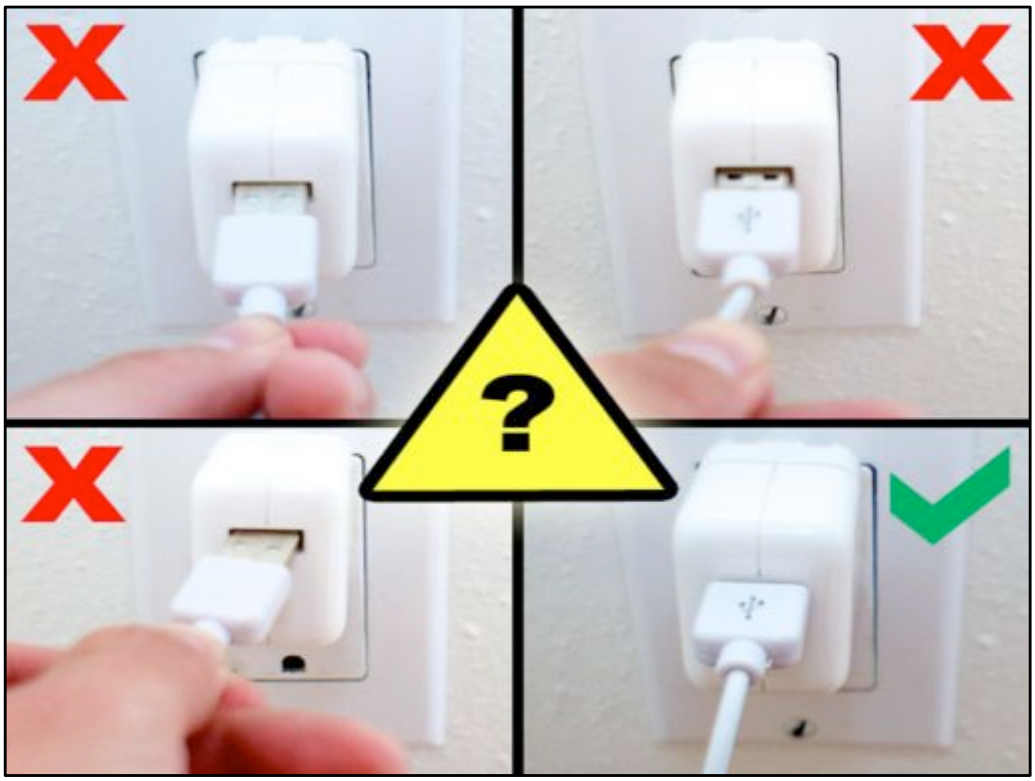
 # Pragmatisme

Le pessimiste se plaint du vent ; L'optimiste attend qu'il change ; Le pragmatique règle des voiles

Laisser de côté la littérature et les définitions de REST ou HATEOAS. Ce n'est pas ma cible.

Garder la philosophie, mais chercher d'abord les cas pratiques et les retours d'expérience
Voir par exemple https://github.com/WhiteHouse/api-standards

***



 # Ne pas refaire les erreurs

Certains ont essayé plein de choses, se sont plantés.

Vous vous planterez aussi, mais autant récupérer ce qui a fonctionné

Moi aussi. Des erreurs de débutant même.

Collection de plein de recettes ou erreurs passées, pour permettre que chacun en fasse de nouvelles au lieu de reproduire les mêmes

 # Objectif
L'idée c'est de ne pas se perdre dans ses objectifs.

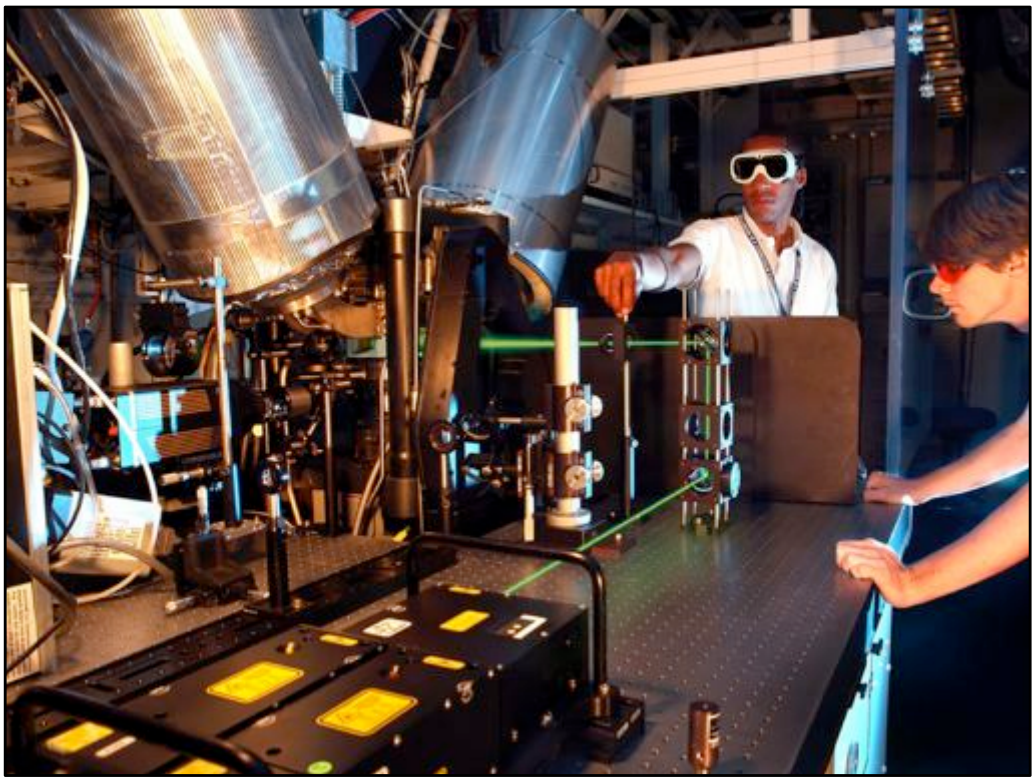
Objectifs :
- Simple
- Rapide à essayer
- Rien à apprendre
- Peu de dépendances
- Sans surprise
- Ne bloque pas les usages non prévus
- Peu de maintenance
- Évolutif
- Sans réinventer la roue
- (à chacun d'ajouter les siens)

Ce qui est dans la littérature mais ne sert pas directement un de ces objectifs, aujourd'hui, peut être laissé de côté

***

 # Opinion

Beaucoup d'opinion dans ce qui suit.
D'autres ont des idées différentes
Objectif : lire, confronter, faire sa propre liste



 # Plans d'architecture complexes irréalistes

OK,
On fait des supers plans, mais bon, le client ne se mettra pas en mesure de comprendre
Il ne lira pas la documentation, n'aura pas forcément les compétences, le temps, la volonté…
Bref, si vous pensez à cette jolie image, vous rêvez

***

http://www.flickr.com/photos/sandialabs/6288974689/ par Sandia Labs sous CC-BY-NC

 # Casse gueule

API : sujet très casse gueule, facile de se planter

http://www.flickr.com/photos/anselmhook/2746893327/ par Anselm Hook sous CC-BY

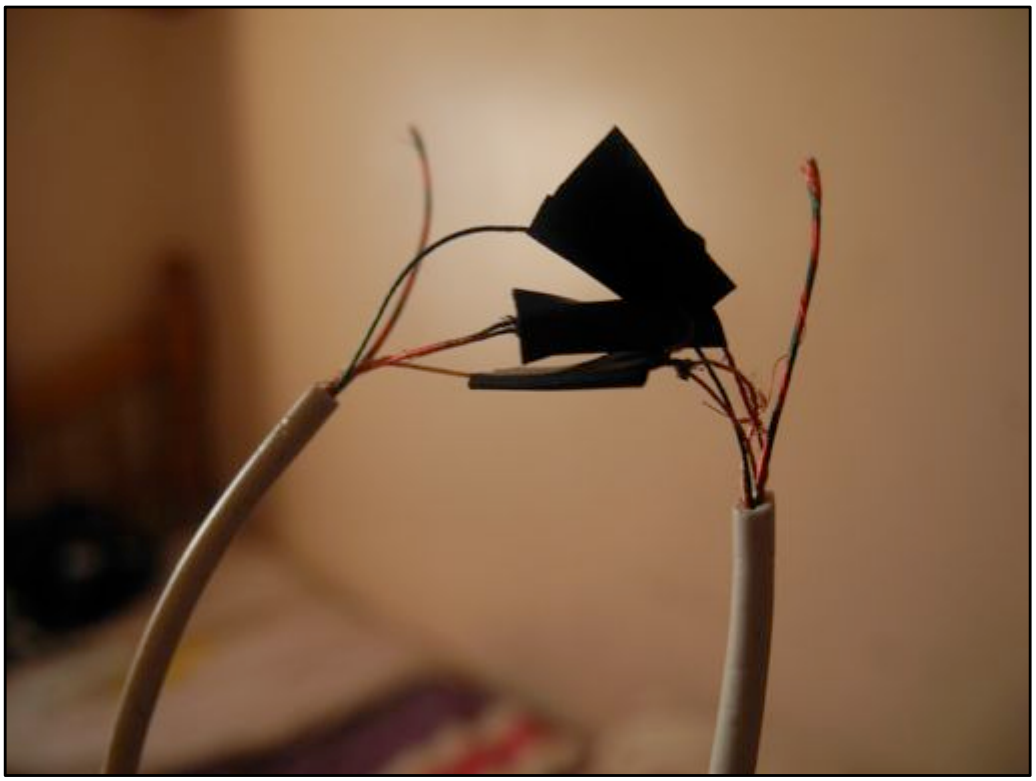
 # Tester

Faire des exemples
Produire un ou plusieurs SDK
Chercher une prise en main rapide
Fournir un bac à sable (API en lecture seule ou avec un jeu de données de test, etc.)
Générateur d'exemples : voir twilio (demande de s'inscrire pour y accéder)

Des tests de recette que vos clients peuvent exécuter en automatique pour valider leur implémentation
(et vous refuser le support tant que leur implémentation ne passe pas)

***

http://www.flickr.com/photos/clintjcl/5940857187/ par Rev. Xanatos Satanicos sous CC-NC-SA



 # Bidouiller

URL en caractères évidents (a-z, 0-9, -, /, rien d'autres, surtout par de majuscules ou de %xx)
URL bidouillables (on doit pouvoir remonter les « répertoires », changer des identifiants, des valeurs de paramètres)
URL qui ressemblent à des fichiers (possiblement sans extension)
Pas ou peu de caché (d'entête) ou de complexe
Paramètres stables sur toute l'API

***
http://www.flickr.com/photos/hellocatfood/5799842139/ par Antonio Roberts sous CC-BY-NC-SA

 # Confiance

Prévoir un jeu de tests exécutables
Ex: pointer vers un mauvais certificat, vérifier qu'il y a erreur

***

http://www.flickr.com/photos/thompsonrivers/6386224021/ de Thompson Rivers University sous CC-NC-SA

 # Vous

***

http://www.flickr.com/photos/skifatenum/3519043883/ par skifatenum sous CC-BY-NC-SA

 # Simple

Faire le minimum, strict nécessaire
Ressources basiques
Peu d'options
De l'obligatoire, pas d'optionnel

***

 # Ne pas se poser de questions

Paramètres génériques : Un paramètre doit être si possible valable pour toute l'API, toutes les ressources, toujours la même sémantique, avec des noms génériques

Noms de ressources spécifiques : éviter les termes génériques genre « contenu », « objet », « actualité », chercher le terme le plus précis possible

Règles nommages stables, éviter les mix pluriel/singulier ou mixCase/under_score, même si vous pensez avoir une règle évidente pour savoir quoi appliquer

Identifiants uniques, simples

***

http://www.flickr.com/photos/skrubu/5061670177 par Pekka Nifrus sous CC-NC-SA

 # Prévoir l'avenir

Structure à plat pour les URIs
Trois niveaux maximum :
/collection
/collection/raccourci
/collection/filtre
/collection/ressource
/collection/ressource/représentation
/collection/ressource/liens
/collection/ressource/sous-element-unique

/collection/ressource/sous-collection/sous-ressource -> /sous-collection/sous-ressource

Identifiants uniques texte : Permet d'envisager plus tard des cas tordus avec des préfixes, suffixes ou règles complexes (on peut toujours l'éviter, et il faut l'éviter, mais s'en réserver la possibilité pourra épargner des problèmes plus tard). Ca vaut parfois la perte de performance

En faire le moins possible

 # Les problématiques techniques plus concrètes

***

http://www.flickr.com/photos/90585146@N08/8234225693/ par marsmetn tallahassee sous CC-BY-NC-SA

 # International

Avec fuseau horaire
Pas de défaut dans votre locale (évite les erreurs/oublis)
Pas de « par défaut en France »

Pas de date, toujours ajouter l'heure (une date ne commence pas partout au même moment)

Au format iso (AAAA-MM-DD) et pas en locale humaine

Pas de localisation des URL

Les messages doivent être des collections, un item par langue, peut être dans une ressource dédiée

Codage UTF8

***

 # Identifiants

Sujets simples, mais pas tant que

Utilisez les votres, tout change (même n° sécu), jamais se baser sur un identifiant tiers en interne

Prévoir la possibilité de changer

Utiliser du texte permet de rester souple

Attention aux bornes (quel maximum ? Cf twitter et ses identifiants qui ont explosé les 32bits

Attention à la prédictibilité (qui peut être parfois gênante pour le commerce)

Evitez les contrainte de taille

UUID ?

Ce qui est unique ne l'est souvent pas

Pensez aux pays par exemple, dont le libellé change, le nom change, etc.

 # Pagination

Semble simple, est complexe
Attention aux perfs, limiter les options, filtres, tris

From / since, pas de notion de page : la collection peut changer entre deux requêtes sur deux pages consécutives

Paramètre obligatoire

Penser au cache

***

http://www.flickr.com/photos/designwallah/4885177922/ par Francis Mariani sous CC-NC-ND

 # Versionnement

En théorie non. En pratique si. Tout le monde en a eu besoin, sans exception

Résistez le plus longtemps possible
Enrichissement sur la même version, compatibilité arrière

Quand vous cédez (et vous le ferez) :
Version majeures uniquement
Pas de version mineure (faites le en enrichissant et avec compatibilité arrière, donc sans besoin de l'annoncer avec des URL ou une mise à jour des clients)

Comment :
Dans la racine de l'URL : /v1/xxxx
À mettre dès la v1

Au pire ça sera juste 3 caractères si vous ne l'utilisez pas dans une v2

***

 # Authentication

HTTP Basic (+tls)
Oauth si besoin. Griller tiers tout service qui demande un mot de passe
Fuyez le custom
Par token, qui sont gérables par l'utilisateur
Ninjas : certificats TLS clients, mais personnellement je n'ai jamais réussi à avoir des clients assez smart pour utiliser ça, même dans des très grosses boites très très sérieuses

***
http://www.flickr.com/photos/janisbrass/8078718025/ par Memaxmarz sous CC-BY-NC-ND

 # Sécurité

TLS par défaut
Ne permettez pas de désactiver la vérification tls dans le SDK
DoS : limitez les profondeurs des requêtes
Limitez la fréquence / par ip ou api_key
Imposez une API_key : vous saurez qui fait les requêtes qui posent problème dans votre API

***

http://www.flickr.com/photos/thomashawk/3147786573/ par Thomas Hawk, sous CC-BY-NC

Erreurs

Codes standards HTTP,
uniquement le code, pas de personnalisation du message
Entièrement, bon investissement pour l'avenir

Code unique incrémental, sur tous les projets, documentés dans un WIKI

Différencier les messages machines et les messages humains
Attention à la localisation

Format

Jetez tous vos préjugés ; Jetez le hype ; Jetez l'habitude

JSON, XML : pas de diff de perf

JSON : plus à la mode, aucune métadonnées, aucune extensibilité, peu ou pas de format établi
XML : un peu moins simple à lire, mais surtout moins à la mode ; Mais attributs, espaces de noms, plein de formats standards, moins se prendre les pieds dans le tapis

HTML ? Oui en théorie avec rdfa ; Mais complexe à relire ;
Reprend l'idée d'un site commun API et humain ; ROI de l'effort pour y arriver ?

Privilégiez le format natif de vos données, mais évitez les formats persos ou le CSV (peu extensible)
Pensez au www-form-urlencoded pour l'entrée

Utilisez des objets/hash et non vos données directes : extensibilités, métadonnées
Ajouter des liens quand c'est simple, mais ne vous prenez pas la tête, les gens

 # Performance

Limite en nombre de résultats
Limiter la profondeur des résultats (n° de page)
Limiter la fréquence des appels
Gzip

 # Cache

Penser au cache
If-modified-since
Cache-control
Etag
Éviter les paginations simples

***

http://www.flickr.com/photos/saralamond/2312090308/ par Sarah Lamond sous CC-BY-NC-ND

 # Accès

GET POST PUT DELETE
PATCH
LOCK

WebDAV

Surcharge via paramètre
Ne pas corrompre la sémantique, au moins GET/POST

JSON-PATCH

***
http://www.flickr.com/photos/synx508/279300958/ par synx508 sous CC-BY-NC

 # Documentation

SDK
Apiary.io
Ne sera pas lue, pas à jour, pas complète
À faire, mais un WIKI sera plus efficace si vous avez des exemples, sdk et jeux de tests de recette

 # Hypermedia

Faites des liens
Par exemple la pagination
Mais n'oubliez pas que les gens ne découvrirons pas l'api par là
Ou trop peu de gens pour que ce soit pertinent
Et impose un problème de performance

***

EN FAIRE PEU
MAIS EN OUVRANT LES POSSIBLES
ENRICHIR DANS UN SECOND TEMPS

SIMPLE, STANDARD,
RESTER PRAGMATIQUE

Éric Daspet
http://eric.daspet.name/

TEA, the ebook alternative
http://www.tea-ebook.com/

Photos sous licence Creative Commons:

http://www.flickr.com/photos/joits/439556240 par Joits sous CC-NC
http://www.flickr.com/photos/tartanna/6364536541/ par Anna Fuster sous CC-NC-ND
http://www.flickr.com/photos/72213316@N00 par Frank Kovalchek sous CC-BY
http://www.flickr.com/photos/anselmhook/2746893327/ par Anselm Hook sous CC-BY
http://www.flickr.com/photos/sandialabs/6288974689/ par Sandia Labs sous CC-BY-NC
http://www.flickr.com/photos/clintjcl/5940857187/ par Rev. Xanatos Satanicos sous CC-NC-SA
http://www.flickr.com/photos/hellocatfood/5799842139/ par Antonio Roberts sous CC-BY-NC-SA
http://www.flickr.com/photos/thompsonrivers/6386224021/ de Thompson Rivers University sous CC-NC-SA
http://www.flickr.com/photos/skifatenum/3519043883/ par skifatenum sous CC-BY-NC-SA
http://www.flickr.com/photos/joyoflife/245379951/ par Julie Kertesz sous CC-BY-NC-SA
http://www.flickr.com/photos/skrubu/5061670177 par Pekka Nifrus sous CC-NC-SA
http://www.flickr.com/photos/leebennett/3181855130/ by Lee Bennett sous CC-BY-NC-SA
http://www.flickr.com/photos/90585146@N08/8234225693/ par marsmetn tallahassee sous CC-BY-NC-SA
http://www.flickr.com/photos/leamarzloff/3204021240/ par Lea Marzloff sous CC-BY-NC-ND
http://www.flickr.com/photos/designwallah/4885177922/ par Francis Mariani sous CC-NC-ND
http://www.flickr.com/photos/jmr-holdit/3008356579/ par J. Michael Raby sous CC-BY-NC-SA
http://www.flickr.com/photos/janisbrass/8078718025/ par Memaxmarz sous CC-BY-NC-ND
http://www.flickr.com/photos/thomashawk/3147786573/ par Thomas Hawk, sous CC-BY-NC
http://www.flickr.com/photos/42386632@N00/8528725328/ par Lauren Macdonald sous CC-BY-NC-SA
http://www.flickr.com/photos/saralamond/2312090308/ par Sarah Lamond sous CC-BY-NC-ND
http://www.flickr.com/photos/synx508/279300958/ par synx508 sous CC-BY-NC
http://www.flickr.com/photos/mctumshie/8016134432/ par Andrew Smith sous CC-BY-NC-ND